Serial No.

University of Bahrain
College of Information Technology
Department of Computer Science
Semester II, 2015-2016
ITCS102/ITCS104/ITCS112 (Computer Programming II)

TEST 1

Date: Tue 5th April 2016 Time:6:00pm - 7:00pm

STUDENT NAME	KEY
STUDENT ID	
SECTION	

QUESTION #	MARKS		COMMENTS
1-PART A	5		
1-PART B	7		
1-PART C	8		
2-PART A	4		
2-PART B	8		
2-PART C	8		
TOTAL	**40**		

Question 1

PART A [5 Points] – String Type

Show the output of the following C++ program:

```
#include <iostream>
#include <string>
using namespace std;
int main( )
{
string X="Rainy or Sunny Day";  string Y;

int L = X.find('n', 5);

cout<<L<<endl;

Y = X.substr(9,3);

cout<<Y<< endl;

Y = Y + "Day";

cout<<Y<<endl;

cout<< X.length()<<endl;

if( X.find('n') <= Y.find('n') )
  cout<<X<<endl;
else
  cout<<Y<<endl;

return 0;
}
```

OUTPUT

// each 1 pt

11
Sun
SunDay
18
SunDay

Question 1

PART B [7 Points] - Arrays

Write a function named **linearSearch** that takes as parameters: an array of integers (list), the number of elements (len), and an integer (x). The function should search sequentially for the element (x) in the array. The function should return the index of the first occurrence of (x) in the array, otherwise the function should return -1. The function prototype is: *int linearSearch(int list[ ], int len, int x);*

```
int linearSearch(int list[], int n, int x)
{

for (int i=0; i<n; i++) // 1 pts

   if(list[i] == x)  // 2 pts

      return i;  // 2 pts

return -1; // 2 pts
}
```

PART C [8 Points]- Arrays

Write a C++ function named **commonElements** that accepts two arrays of integers, array1 and array2 and their sizes len1 and len2. The function should count and return the number of common numbers in both arrays. Call the function **lineraSearch** in PART (B) to write the function **commonElements**.

For example, if array1={30,40,50} and array2={40,22,50,55}; the function should return 2 (number of common numbers in both arrays).

The function prototype is: *int commonElements(int array1[ ], int array2[ ], int len1, int len2);*

```
int commonElements(int array1[], int array2[], int n1, int n2){

int count = 0; // 1 pt

for(int i=0; i<n2; i++) // 2 pts

 if(linearSearch(array1, n1, array2[i]) != -1) // 3 pts

     count++; // 1 pt

return count; // 1 pt

}
```

Question 2 [Struct]

PART A [4 Points]

Define a struct named **dateType**, which include the following members: day, month and year. For example 5/4/2016.

PART B [8 Points]

Define a struct named **sweetType** to represent a sweet sold in a shop, which include the following:

1. **Name**: to represent the sweet name.
2. **ExpDate**: to represent the Expiry date of the Sweet (type DateType)
3. **Type**: to represent the type of the sweet: ex Cake, pie, cookies…
4. **ING**: an array of Maxsize 10 to represent the names of main ingredient of the sweet.
5. **No**: number of the ingredient.

For example:

Name:	Apple Pie				
ExpDate:	6/5/2016				
Type:	Pie				
ING:	Flour	Sugar	Water	Apple	Cinimmon
No:	5				

```
struct dateType{ // 1 pt
      int day; // 1 pt
      int month; // 1 pt
      int year; // 1 pt
};


struct sweetType{ // 1 pt
  string Name; // 1 pt
  dateType ExpDate; // 2 pt
  string Type;  // 1 pt
  string ING[10]; // 2 pt
  int No; // 1 pt
};
```

Question 2 [Struct].. Continue

PART C [8 Points]

Write a function named isPieExpired that takes one parameter: ***sweet*** of type sweetType. The function should return *true* if the year of ExpYear is 2015 and type is "Pie", otherwise the function should return false. The function prototype is:

bool* isPieExpired *(sweetType sweet);

```
// check year   = 3 pt
// check type  = 3 pts
// return true if both conditions valid = 1 pt
// return false = 1 pt

bool isPieExpired (sweetType sweet){

    return(sweet.ExpDate.year == 2015 && sweet.Type == "pie");
}
```